MICROCONTROLLER BASED IMPLEMENTATION OF A FUZZY KNOWLEDGE BASED CONTROLLER

DEBASMITA PATTNAIK (109EE0298)

BONANI SAHU (109EE0302)

DEVADUTTA SAMANTARAY (109EE0061)

DEPARTMENT OF ELECTRICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

MICROCONTROLLER BASED IMPLEMENTATION OF A FUZZY KNOWLEDGE BASED CONTROLLER

A Thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in "Electrical Engineering"

By

DEBASMITA PATTNAIK (109EE0298)

BONANI SAHU (109EE0302)

DEVADUTTA SAMANTARAY (109EE0061)

Under guidance of

Prof. SUBHOJIT GHOSH

Department of Electrical Engineering

National Institute of Technology

Rourkela-769008 (ODISHA)

May-2013

DEPARTMENT OF ELECTRICAL ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA- 769 008
ODISHA, INDIA

CERTIFICATE

This is to certify that the draft report/thesis titled "MICROCONTROLLER BASED IMPLEMENTATION OF A FUZZY KNOWLEDGE BASED CONTROLLER", submitted to the National Institute of Technology, Rourkela by **Bonani Sahu, Roll No**: **109EE0302, Debasmita Pattnaik, Roll No: 109EE0298 & Devadutta Samantaray, Roll No: 109EE0061** for the award of Bachelor of Technology in Electrical Engineering, is a bonafide record of research work carried out by them under my supervision and guidance.

The candidates have fulfilled all the prescribed requirements.

To my knowledge, the draft report/thesis which is based on candidates' own work, has not submitted elsewhere for a degree/diploma.

In my opinion, the draft report/thesis is of standard required for the award of a Bachelor of Technology in Electrical Engineering.

Prof. Subhojit Ghosh

Supervisor

Department of Electrical Engineering

National Institute of Technology

Rourkela – 769 008 (ODISHA)

ACKNOWLEDGEMENT

We have been highly indebted in the preparation of this report to our supervisor, Prof. Subhojit Ghosh, whose patience and kindness, as well as his academic experience, has been invaluable to us.

The informal support and encouragement of many friends has been indispensable. We would not have contemplated this road if not for our parents, who instilled within us a love of creative pursuits, science and language, all of which finds a place in this report.

ABSTRACT

In recent times, fuzzy logic has been used and applied in wide areas, starting from consumer electronics like washing machines to robotics to many industrial control systems like temperature controllers for process plants.

Our work describes an implementation of fuzzy logic control algorithm using inexpensive hardware to control the temperature of a system, without any special software tools. A cooling system generally involves complex and time-variant plant, with delays and non- linearity, and often with poorly defined dynamics. Fuzzy logic control algorithm solves problems that are difficult to address with traditional control techniques, and at the same time provides us with a response better than conventional PID controllers. In the present work, this has been proved with the help of MATLAB simulations.

Thereafter the program for the fuzzy control algorithm is written in C++ language and implemented through ARDUINO UNO tool kit. Further system functional is tested and the performance is evaluated taking several set-points and disturbances into account. The performance of the hardware is compared with that of MATLAB simulations of the same case and the results are verified.

CONTENTS

CHAPTER-5

EXPERIMENTAL RESULTS AND COMPARITIVE STUDY

CHAPTER-6

CONCLUSION AND FUTURE WORK

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

AC	-	Alternating Current
DC	-	Direct Current
PWM	-	Pulse Width Modulation
EMI	-	Electro Magnetic Interference
MATLAB	-	MATrix LABoratory
PID	-	Proportional, Integral and Derivative
FLC	-	Fuzzy Logic Controller
MF	-	Membership Function
CPU	-	Central Processing Unit
USB	-	Universal Serial Bus
FTDI	-	Future Technology Devices International
IC	-	Integrated Circuit
LED	-	Light Emitting Diode
SMPS	-	Switched Mode Power Supply
SPI	-	Serial Peripheral Interface

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION:

Nowadays, in globalization era there are always the foundation of the new technologies features every year. Automatic control system has become the most popular feature which is rapidly gaining its popularity due to its importance to certain applications. Process control systems are often nonlinear and difficult to control accurately and efficiently. These dynamic models are more difficult to derive than those used in aerospace or robotic control, and they tend to change in an unpredictable way. The conventional PID controllers, in various combinations have been widely used for industrial processes due to their simplicity and effectiveness for linear systems, especially for first and second order systems. It has been well known that Proportional Integral Derivative (PID) controllers can be effectively used for linear systems, but usually cannot be used for higher order and nonlinear systems. Fuzzy logic has emerged as one of the active areas of research activity particularly in control applications. It is a very powerful method of reasoning when mathematical models are not available and input data are imprecise. Fuzzy logic performs better when compared to conventional control mechanisms like PID. The main objective behind fuzzy logic is to represent and reason with some particular form of knowledge expressed in linguistic form.

This work addresses an application that involves the system control system. It presents a fuzzy controller that uses an adaptive neuro-fuzzy inference system. Fuzzy Inference system (FIS) is a popular computing framework and is based on the concept of fuzzy set theories, fuzzy if and then rules, and fuzzy reasoning.

1.2 LITERATURE REVIEW:

Implementation of fuzzy logic technology for the development of sophisticated control systems has become one of the most rapidly growing successful technologies. This is mainly because fuzzy logic resembles human decision making with an ability to generate precise solutions from approximate information. It fills up an important gap in engineering design methods left vacant by purely mathematical approaches (e.g. linear control design), and purely logic-based approaches (e.g. expert systems) in system design. While other approaches require precise equations to model real-world behaviours, fuzzy design can accommodate the uncertainties of real-world human language and logic by providing both an intuitive method for describing systems in human terms and automating the conversion of those system specifications into effective model [1]. Fuzzy Logic relates input to output in linguistic terms that can be easily understood. It allows for the rapid prototyping because the

system designer doesn't need to know everything about the system before starting. It is cheaper because they are easier to design. It involves simplified knowledge acquisition and representation. It is used to achieve less overshoot and oscillation. It achieves steady state in a shorter time interval.

Fuzzy logic was coined in the year 1965 by Lotfi Zadeh. From then on, the history of fuzzy logic follows the pattern of many recent key technologies: invented in the U.S., engineered to perfection in Europe, and currently, mass-marketed in Japan.

The use of fuzzy logic provides very fast response and reliable operation. As the software is more or less common for all control application, we can use this fuzzy control for other applications including non-linear systems. The ultimate advancement possible will be incorporation of neural networks in combination with the fuzzy algorithm. Neural networks can be used to absolutely authorize the design process of fuzzy systems [1]-[3]. The FLC performance is superior to the PID controller, presenting faster transient response and less overshoot and oscillation. It is also more robust against disturbances than the PID controller. But it has some steady state errors due to the coarse tuning and small size of fuzzy set which can be achieved by resizing the fuzzy sets and finer tuning for the membership functions [4]-[8]. Unlike some fuzzy controllers with hundreds and thousands of rules running on computer systems, a unique FLC that uses a small number of rules and simple implementation can be used to solve a system control problem with unknown dynamics or variable time delays commonly found in industry. The control result can be improved by resizing the fuzzy sets and finer tuning for the membership functions [9]. A closed loop control system incorporating fuzzy logic has been developed for a class of industrial temperature control problems. A unique fuzzy logic controller (FLC) structure with an efficient realization and a small rule base that can be easily implemented in existing industrial controllers was proposed. It was demonstrated the potential of FLC in both software simulation and hardware test in an industrial setting. This includes compensating for thermo mass changes in the system that deals with unknown and variable delays, operating at various temperature set-points without retuning, etc. It is achieved by applying, in FLC, a classical control strategy and an adaptation mechanism to compensate for the dynamic fluctuations in the system.

Fuzzy logic not only replaces conventional control techniques, but also provides a solution where conventional methods are not satisfactory. When a present control solution actually exists, replacement of fuzzy logic may not be indispensable. But it is not often true.

An alternative solution by using Fuzzy logic control may be better. It all depends on how far the system under control is known to us in its parameters, variables and various relationships of control. If determined values of such variables are not existing, then fuzzy logic based classification of the variables provides a solution which may be better than a method of control using\assumed relationship.

1.3 THESIS OBJECTIVES:

The following objectives are hopefully to be realized at the end of the project.

1) To study the fuzzy logic and its characteristics.
2) To study the comparison in the performances of FLC and conventional control algorithms like PID.
3) To develop a controller which could take in an analog signal, compare it with a certain reference value and then give the control output required for the signal to reach the reference.
4) To authenticate experimental results obtained from the laboratory set-up and to analyse the results with the simulated results in the MATLAB-Simulink Environment.

1.4 ORGANISATION OF THESIS:

The thesis is organised into six chapters including the chapter of introduction. Each chapter is different from the other and is described along with the necessary theory required to comprehend it.

Chapter 2 deals with an overview of Fuzzy Logic. It describes the inception of fuzzy logic, followed by a description of the types of membership functions used in fuzzy sets. The layout of a Fuzzy Inference system is also described and its various procedures are explained.

Chapter 3 describes the analysis and design of Fuzzy Logic Controller. The choice of a certain rule base used is justified and a FLC is designed using this particular rule base. The performance of FLC is compared with that of conventional control algorithms, taking PID as an example. The comparison is done in a MATLAB-Simulink environment and the result is verified.

Chapter 4 shows the practical implementation of the proposed model. The hardware used for the implementation is briefly described, followed by a study of all the blocks required in the actual model. An ARDUINO hardware model and a MATLAB-Simulink model of the controller block are simultaneously designed. Then, the overall experimental setup is charted out for open loop configuration of the controller.

Chapter 5 presents the experimental results of the hardware block so designed. The output waveform generated from the hardware is compared with that of the Simulink model to verify the results.

Chapter 6 concludes the work performed so far. The possible limitations in proceeding research towards this work are discussed. The future work that can be done in improving the current scenario is mentioned. The future potential along the lines of this work is also discussed.

CHAPTER 2

OVERVIEW OF FUZZY LOGIC

2.1 INTRODUCTION

Lotfi Zadeh, the father of fuzzy logic, claimed that many sets in the world that surrounds us are defined by a non-distinct boundary. Zadeh decided to extend two-valued logic, defined by the binary pair {0, 1}, to the whole continuous interval [0, 1], thereby introducing a gradual transition from falsehood to truth. *Fuzzy control* is a control method based on *fuzzy logic*. Just as fuzzy logic can be described merely as "computing with words rather than numbers"; fuzzy control can be described simply as "control with sentences rather than equations". A fuzzy controller can include empirical rules that are mainly useful in operator controlled plants.

Even though the broad sense of fuzzy logic covers a wide range of theories and techniques, its core technique is based on four basic concepts:

(1) Fuzzy sets: sets with smooth boundaries;

(2) Linguistic variables: variables whose values can be described both qualitatively and quantitatively by fuzzy sets;

(3) Possibility distribution: constraints on the value of a linguistic variable imposed by assigning it a fuzzy set; and

(4) Fuzzy if then rules: a knowledge representation scheme for describing a functional mapping or a logical formula that generalizes an implication in two-valued logic.

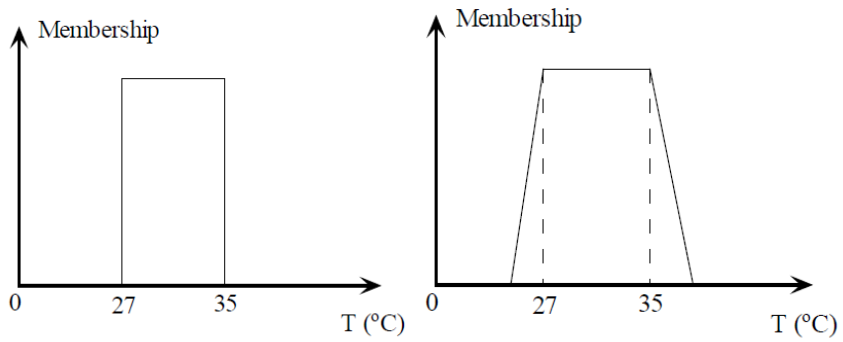


FIGURE 2.1: (a) CLASSICAL SET (b) FUZZY SET

2.2 MEMBERSHIP FUNCTIONS

2.1.1 DEFINITION

A fuzzy set is defined by a function that maps objects in a domain of concern to their membership value in the set. Such a function is termed as membership function and is usually

denoted by the Greek symbol μ for ease of recognition and consistency. A membership function can be designed in three ways:

(1) Interview those who are familiar with the underlying concept and later adjust it based on a tuning strategy.

(2) Build it automatically from data;

(3) Learn it on the basis of feedback from the system performance.

2.1.2 TYPES OF MEMBERSHIP FUNCTIONS

a) Triangular Membership Function

This MF is mainly classified by three parameters namely {a,b,c}

So it can be represented by

$$\text{Triangle}(x;a,b,c)= \begin{cases} 0, & x \le a \\ (x-a)/(b-a), & a \le x \le b \\ (c-x)/(c-b), & b \le x \le c \\ 0, & c \le x \end{cases}$$

By using min and max, we have alternative expression as follows

$$\text{Triangle}(x,a,b,c)=\max(\min((x-a)/(b-a), (c-x)/(c-b)),0)$$

Where the parameters a,b,c determine the x coordinates of the three corners of the underlying triangular MF.

b) Trapezoidal Membership Function

This MF is mainly classified by four parameters namely {a,b,c,d}

So it can be represented by

$$\text{Trapezoid}(x;a,b,c,d)= \begin{cases} 0, & x \le a \\ (x-a)/(b-a), & a \le x \le b \\ 1, & b \le x \le c \\ (c-x)/(c-b), & c \le x \le d \\ 0, & d \le x \end{cases}$$

By using min and max, we have alternative expression as follows

$$\text{Trapezoid}(x,a,b,c,d)=\max(\min((x-a)/(b-a),1, (d-x)/(d-c)),0)$$

Where the parameters a,b,c,d determine the x coordinates of the three corners of the underlying trapezoidal MF.

c) Gaussian Membership Function

This MF is mainly classified by four parameters namely {c,σ}

$$\text{Gaussian}(x; c,\sigma) = e^{-.5(\frac{x-c}{\sigma})^2}$$

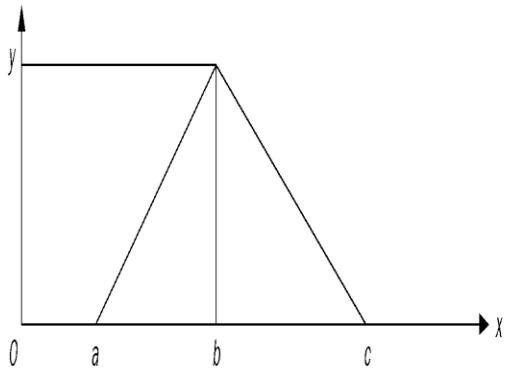
Where c and σ are the centre and width of the MF respectively.

d) Generalised Bell Membership Function

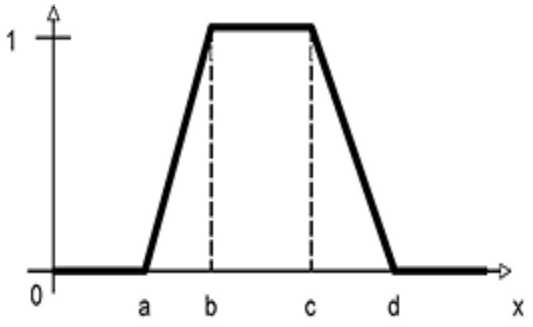
This MF is mainly classified by four parameters namely {a,b,c}

$$\text{bell}(x; a, b, c) = \frac{1}{1+\left(\left|\frac{x-c}{a}\right|\right)^{2b}}$$

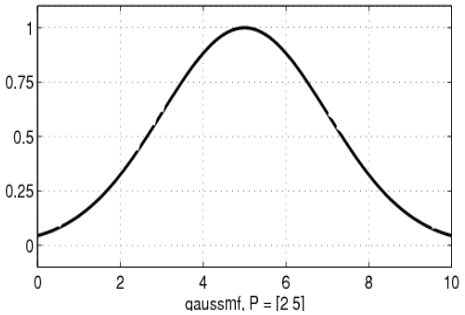
Where b is always positive .We can vary c and a to vary centre and width of the MF and b to control slopes at crossover points.



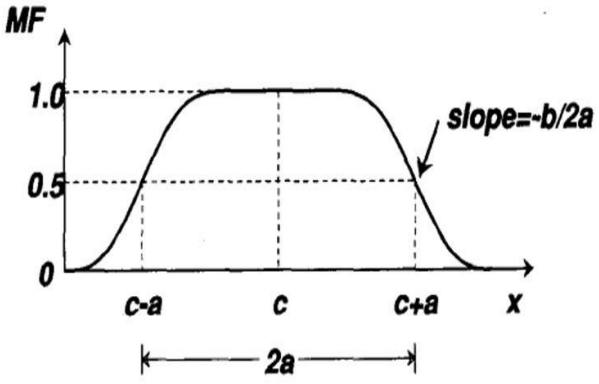
(a)



(b)



(c)



(d)

FIGURE 2.2: DIFFERENT TYPES OF MEMBERSHIP FUNCTIONS: (a) TRIANGULAR (b) TRAPEZOIDAL (c) GAUSSIAN (d) GENERALIZED BELL

2.3 FUZZY INFERENCE SYSTEM

- *Fuzzification*: Converting crisp facts into fuzzy sets described by linguistic expressions. Membership functions can be flat on the top, piece-wise linear and triangle shaped, rectangular, or ramps with horizontal shoulders. These three choices can be explained by the ease with which a parametric functional description of a membership function can be obtained, stored with minimum use of memory and employed efficiently, in terms of real time requirements by the inference engine.
- *Inference*: The fuzzy IF-THEN rule expresses a fuzzy implication relation between the fuzzy sets of the premise and the fuzzy sets of the conclusion. The basic function of the rule base is to represent in a structured way the control policy of an experienced process operator in the form of a set of production rules such as:

If (process state) then (control output)

The "if" part of such a rule is called rule antecedent and is a description of a process state in terms of a logical combination of atomic fuzzy propositions .The "then" part of the rule is called the rule consequent and is again the description of the control output in terms of a logical combination of fuzzy propositions. These propositions represent the linguistic values which the control outputs take when the current process state matches the process state description in the rule antecedent. In our terminology, a fuzzy control rules provide a convenient way for expressing control policy and domain knowledge. The proper choice of process state variables and control variables is essential to the characterisation of the operation of a fuzzy system. Typically the linguistic variables in a FLC are states, state error, state error derivative and state error integral.

- *Aggregation*: This process aggregates the individual rule outputs to obtain the overall system output. It will be also a fuzzy subset over the output universe (a union operation yields a global fuzzy set of the output).

- *De-fuzzification:* to obtain crisp output (various de-fuzzification methods can be used, as, e.g., center of gravity, bisector of area, and mean of maximum, to obtain a crisp numerical output value).

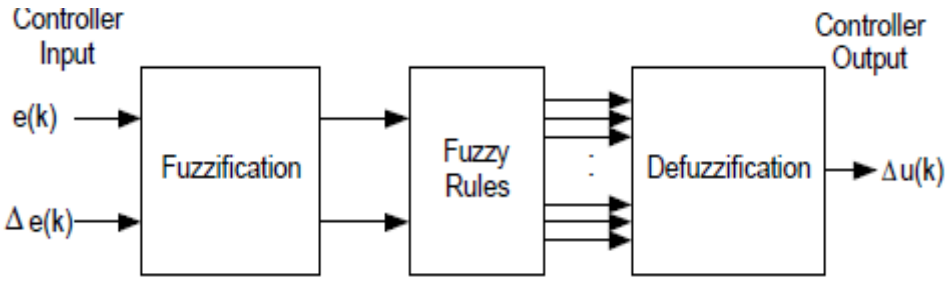


FIGURE 2.3: STRUCTURE OF FIS

Fuzzy logic is a powerful way to put engineering expertise into products in a short amount of time. It's highly beneficial in automotive engineering, where many system designs involve the experience of design engineers as well as test drivers. Due to their simple formulas and computational efficiency, both triangular and trapezoidal MFs have been extensively used especially in real time implementation. However since the MFs are composed of straight line segments, they are not very smooth at the corner points specified by the parameters. Since trapezoidal MF has two corner points while triangular MF has just one, we use triangular MF most frequently among all MFs. The implementation of fuzzy logic for control applications requires a microprocessor or Microcontroller based system. The microcontroller offers low cost and compact digital systems due to integration of CPU, memory devices and peripheral devices in a single chip.

CHAPTER 3

DESIGN AND ANALYSIS OF FUZZY LOGIC CONTROLLER

3.1 INTRODUCTION

Many fuzzy logic systems consist of multiple components. For example, one component may assess a process variable for which a sensor does not exist by utilizing related input signals. Centred on this fuzzy estimation and other inputs, another component can define the actual control strategy. Each component of the fuzzy logic system contains a subset of the complete fuzzy rule set and is thus called a "rule block". To connect rule blocks to each other, some intermediate linguistic variables can be used. Because these variables are never fuzzified or de-fuzzified, no membership functions are required. Linking rule blocks with inputs, outputs, and intermediate linguistic variables define the inference structure of the fuzzy logic system. The rule blocks in the fuzzy logic design contain the actual control strategy. Fuzzy rule design is divided into two steps. Firstly, the formulation of initial rule blocks, and secondly, to optimize the rules based on analysis and testing in the last two design steps.

3.2 CONSTRUCTION OF A RULE BASE TABLE

In a conventional PI controller

$$\mathbf{u(t)}=K_c[e(t)+\frac{1}{T_i}\int e(\tau)d\tau] \quad \textbf{.... (1)}$$

$$\frac{du(t)}{dt}=K_c'[\frac{de(t)}{dt}+\frac{e(t)}{T_i}] \quad \textbf{..... (2)}$$

In discrete time it can be written as

$$\frac{u(k)-u(k-1)}{T}=\frac{K_c'}{T}[e(k)-e(k-1)]+\frac{K_c'}{T_i}e(k) \quad \textbf{..... (3)}$$

$$\Delta u(k)=K_c v(k)+K_i e(k) \quad \textbf{..... (4)}$$

Control action desired:

Δu(k)= 0 in set point region

= very large in constant region

= normal in between

- A negative value of e(k) means that current process output y(k) is greater than yr and vice-versa

- A negative value of v(k) means that current process output y(k) has increased from y(k-1) and vice-versa
- A positive value of Δu(k) means that u(k-1) has to be increased to obtain value of the control for current sampling time k and vice-versa
- If both e(k) and v(k) are small or zero it means that the current value of process output variable y(k) has deviated from the set-point but is still close to it. The amount of change Δu(k) should also be small in magnitude
- If e(k) has large negative value which implies that y(k) is significantly above the set-point. If v(k) is positive at the same time, it means y is moving towards the set-point. The amount of change Δu(k) to be introduced is intended to either speed up or slow down the process.
- If e(k) has a very small value or a large positive value which it implies that y()k is either close to the set point or significantly above it. If v(k) is positive at this time it means that y is moving away from the set point. Thus a positive change Δu(k) is required to reverse the trend.
- If e(k) has a large positive value i.e. y(k) is significantly below the set-point and v(k) is negative. The amount of change Δu(k) will either speed up or slow down the process
- If e(k) has a small value (positive, negative, zero)or a large negative value it means that y(k) is either close to set point or is significantly above it. If v(k) is negative at the same time y is moving away from the set point. Thus a negative change in Δu(k) will reverse the trend.
- From the above inference, the rule base can be made for the PI controller as follows

e \ du	N	NZ	P
N	N	N	Z
NZ	N	Z	P
P	Z	P	P

TABLE 3.1- RULE-BASE TABLE OF PI CONTROLLER

3.3 SIMULATION IN MATLAB

The FLC block and the PID block were created as shown in the figure given below. In this simulation we take a step input as the reference and a model transfer function of $\frac{1}{(1+s)^3}$ as the plant. The controller is taken as a conventional PID controller in one case and a FLC in another case. The block diagram is shown.

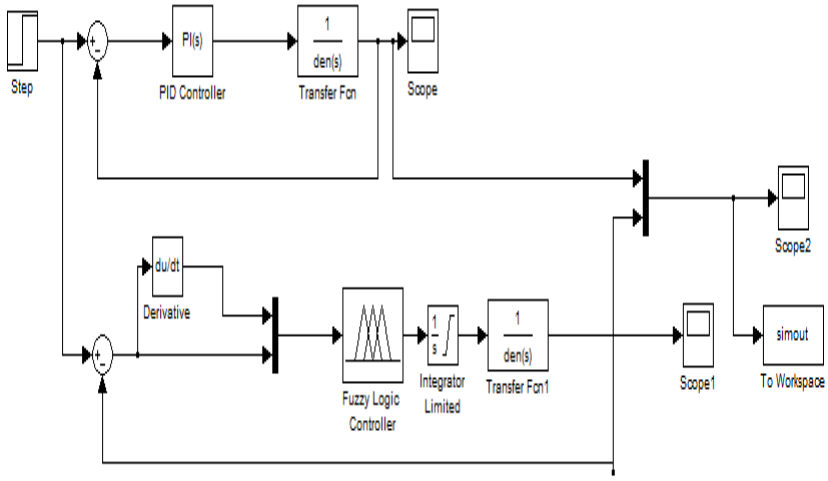


FIGURE 3.1- BLOCK DIAGRAM FOR DESIGN OF THE CONTROLLER

3.4 RESULTS

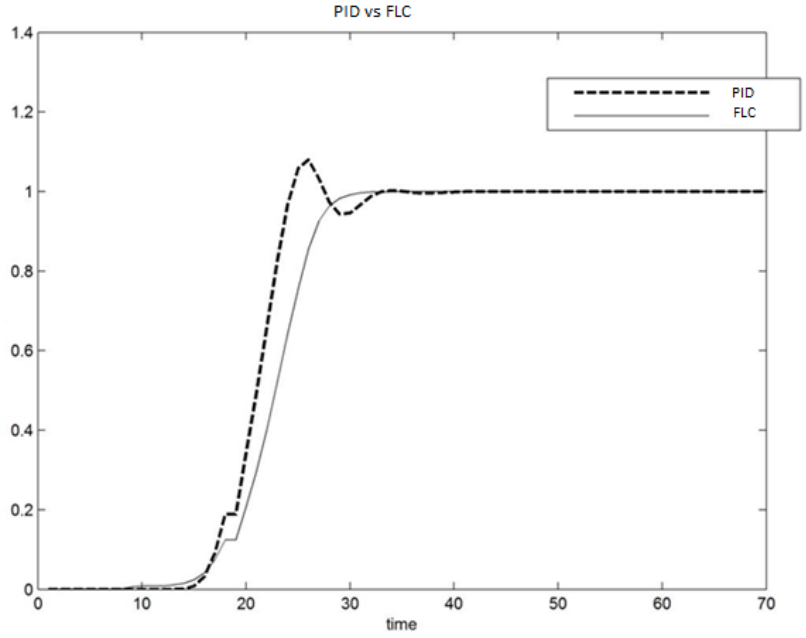


FIGURE 3.2- COMPARISON OF THE PERFORMANCE OF PID AND FLC CONTROLLER FOR A STEP INPUT

3.5 CONCLUSION

From the simulation it was clear that a fuzzy logic control is better than conventional PID control because the settling time of fuzzy logic controller is less than that of the PID controller. Moreover the great advantage of a fuzzy logic controller is that it can be implemented for a non-linear system as compared to a PID controller which is suitable only for a linear system. The fuzzy logic controller can operate even if the transfer function is not given. But while using PID controller we need to have the transfer function. Fuzzy logic controller is very flexible because if there is any change in the poles or zeroes it automatically adapts itself to the change whereas in PID controller we need to tune it according to the new conditions. It is because of all these advantages FLC is much more preferred over PID controller.

CHAPTER 4

HARDWARE IMPLEMENTATION OF FUZZY LOGIC CONTROLLER

4.1 INTRODUCTION

The Fuzzy Logic Controller used has two inputs (error and change in error) and has a single output that is given as Control Input to the desired plant. First, the membership function of the FLC is optimized by using hit and trial method using MATLAB. After optimization, it was found out that after fuzzification and de-fuzzification, the crisp output that we were getting had its range between -7 to 7. But the Digital Microcontroller operates in the **TTL** of 0V to 5V. So, inside the code, the crisp value was converted to the required range.

The data from the optimized membership functions were used in writing the C++ code. The C++ code was converted to Arduino language and was burned in the microcontroller present in the Arduino hardware. The required debugging was done and the final PWM output was seen through a DSO. The PWM output from the hardware was then compared with the PWM output that was found out by simulating in MATLAB.

4.2 COMPONENTS USED:

ARDUINO UNO TOOL KIT

The Arduino Uno is a microcontroller board based on the ATmega328 [11]. It has 14 digital input/ output pins (of which 6 can be used as PWM outputs), six analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything that is needed to support the microcontroller; one can get started by simply connecting it to a computer with a USB cable or powering it with an AC-to-DC adapter or battery. It does not use the FTDI USB-to-serial driver chip, thereby making this Uno different from all preceding boards. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" translating to one in English, is named so to mark the upcoming release of Arduino 1.0.

The Arduino Uno can be powered via the USB connection or with an external power supply [12]. External (non-USB) power can be derived either from an AC-to-DC adapter (wall-wart) or battery by plugging in a 2.1mm centre-positive plug into the board's power jack to complete the connection. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board, operational on an external supply of 6 to 20 volts, might become unstable if supplied with less than 7V, such that the 5V pin supplies less

than five volts. Similarly the voltage regulator may overheat and damage the board when supplied with a voltage more than 12 V. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board during the usage of an external power source (as opposed to 5 volts from the USB connection or other regulated power source). Voltage can be supplied through this pin, or, if supplied via the power jack, it can be accessed through this pin.
- **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with a power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). The board can be damaged by supplying voltage via the 5V or 3.3V pins which could bypass the regulator.
- **3V3.** A 3.3 volt supply generated by the on-board regulator with a maximum current draw of 50 mA.
- **GND.** Ground pins.
- **IOREF.** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. An appropriately configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

***Memory*:** The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output:

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead()functions, each operating at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. Additionally, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the respective pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or a falling edge, or a change in value.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins can support SPI communication using the SPI library.

- **LED: 13.** There is a built-in LED connected to the digital pin 13. When pin is HIGH value, the LED is on, when the pin is LOW, it is off.

The Uno has 6 analog inputs, labeled A0 to A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts. But it is possible to change the upper end of their range using the AREF pin and the analogReference() function.

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using Wire library.

There are several other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** This line is brought LOW to reset the microcontroller.It is typically used to add a reset button to shields which block the one on the board.

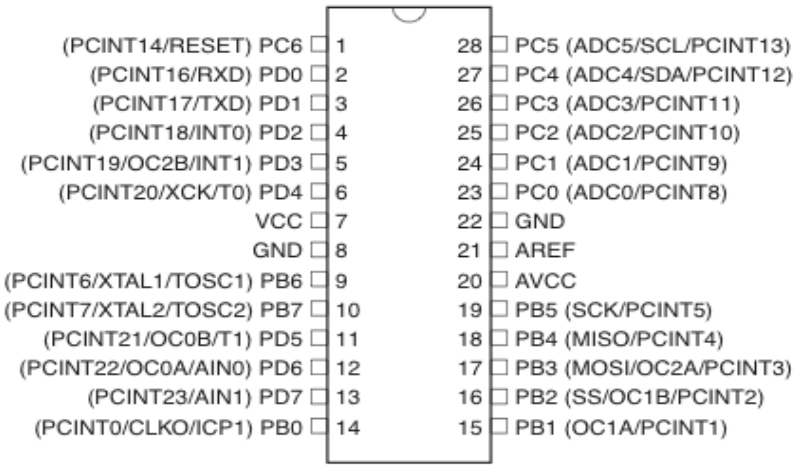


FIGURE 4.1- MICROCONTROLLER ATMEGA 328

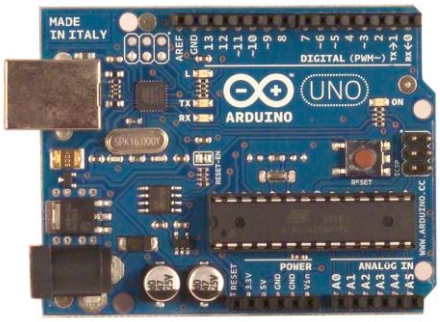


FIGURE 4.2- ARDUINO UNO TOOL KIT

In this hardware op-amp is used as comparator. Now the error from the comparator element is sense in the ADC. ADC converts this analog error signal into digital form. Microcontroller senses this error signal and produces the controlled output in the form of digital data by processing the algorithm in EPROM proportional to error voltage, which generates the digital output. This signal can be sent to a DAC which can produce this controlled data in the form of analog current which is converted to proportional voltage in a current to voltage amplifier.

4.3 EXPERIMENTAL SETUP:

4.3.1 *HARDWARE SETUP*

The whole experimental setup is centred on the Arduino kit (Figure 4.6). It is done in the following steps:

- The positive terminal of function generator is connected with the A3 pin of Arduino.
- The negative terminal of function generator is connected with the GND pin of the hardware.
- The output is taken from the PIN 6 of the Arduino kit and is connected to the DSO through Channel-1. The DSO is then grounded.

After the setup is complete, power is given to the function generator and a step voltage within a range of 0.5V to 2.5V is given as input to the Arduino hardware and the corresponding PWM output is checked in the DSO.

4.3.2 *SIMULINK MODEL SPECIFICATIONS*

MEMBERSHIP FUNCTIONS USED:

The FLC designed in MATLAB used triangular membership functions since they are easy to be calculated (having lesser number of parameters) and easier to modify. Additionally, they are economical for the same reason. The ranges and parameters are shown in the figure given below.

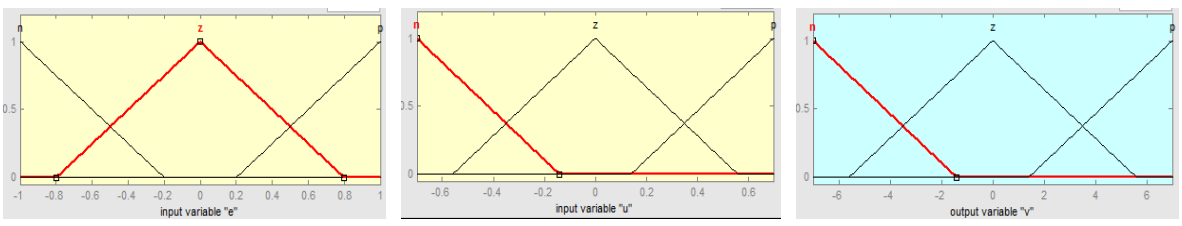


FIGURE 4.3- MEMBERSHIP FUNCTIONS USED IN THE FIS FILE

4.4 MODEL LAYOUT:

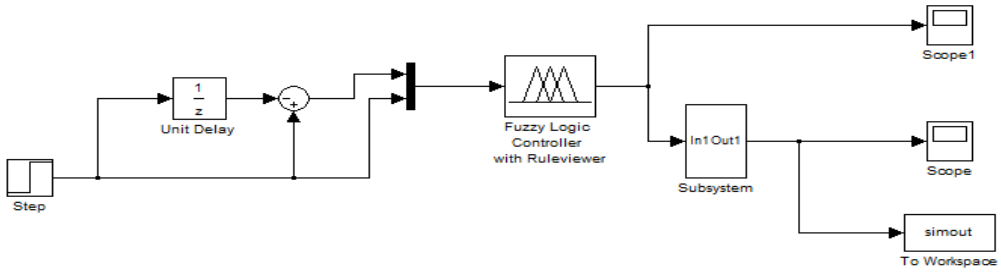


FIGURE 4.4- MATLAB SIMULINK DIAGRAM OF A FUZZY LOGIC CONTROLLER

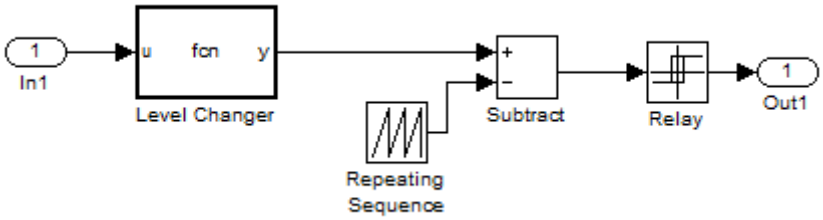


FIGURE 4.5- SCHEMATIC DIAGRAM OF PWM BLOCK

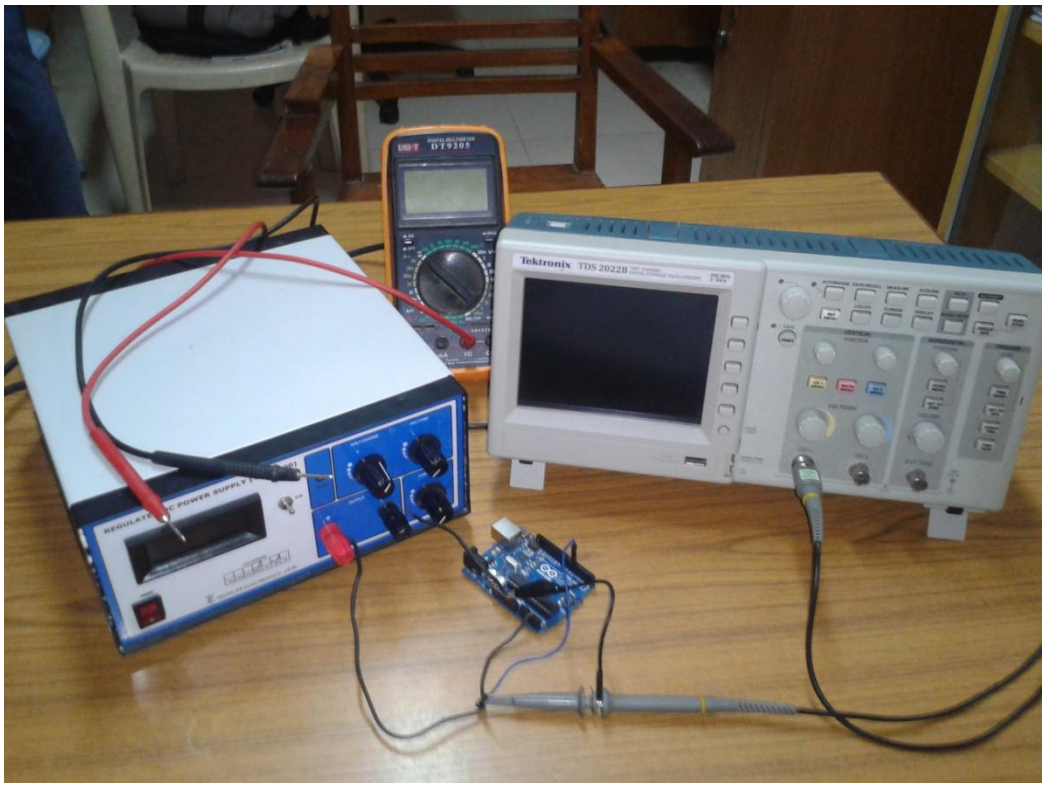


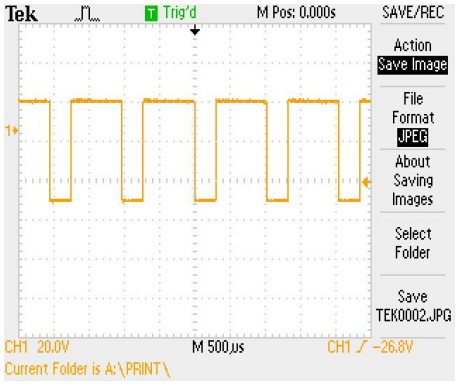
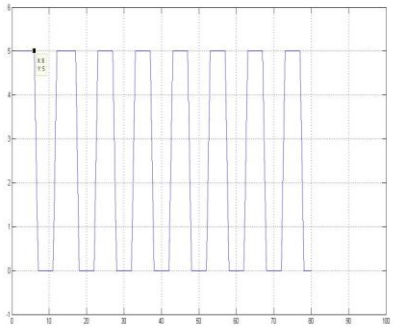
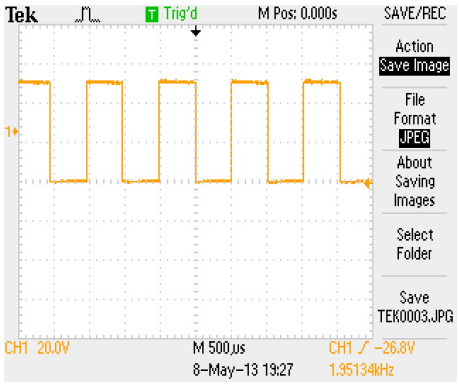
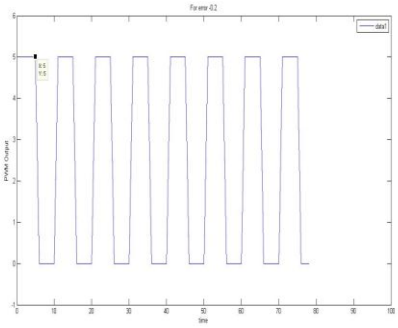
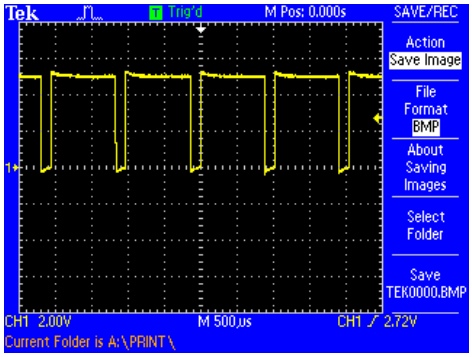
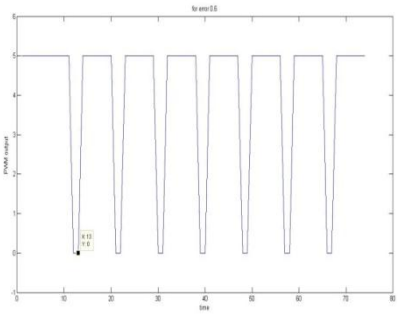
FIGURE 4.6- EXPERIMENTAL SETUP OF THE HARDWARE MODEL OF THE CONTROLLER

CHAPTER 5

EXPERIMENTAL RESULTS AND COMPARATIVE STUDY

5.1 COMPARATIVE STUDY:

In this chapter, a comparative analysis has been carried out between the response of the fuzzy logic controllers implemented in Simulink and Arduino microcontroller kit. The PWM output of the Arduino kit has been captured on a Tektronix Digital Storage Oscilloscope. Table 4.1 reports the responses for different magnitude of the error signal.

Specifications	Arduino Kit Output	Simulink Output
REFERENCE VALUE= 1.5V GIVEN VOLTAGE= 1.1V ERROR=0.4 V		
REFERENCE VALUE= 1.5V GIVEN VOLTAGE= 1.7V ERROR= -0.2 V		
REFERENCE VALUE= 1.5V GIVEN VOLTAGE= 0.9V ERROR= 0.6 V		

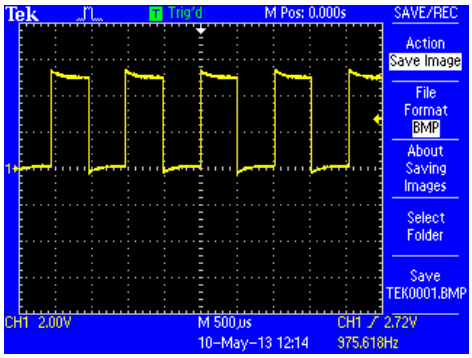
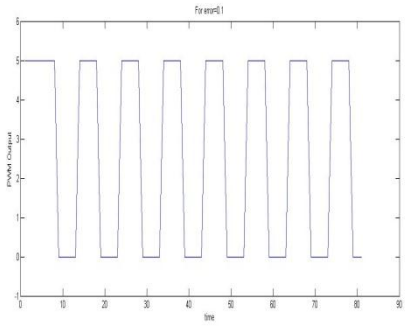
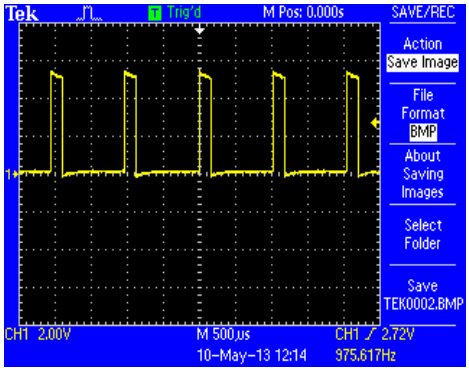
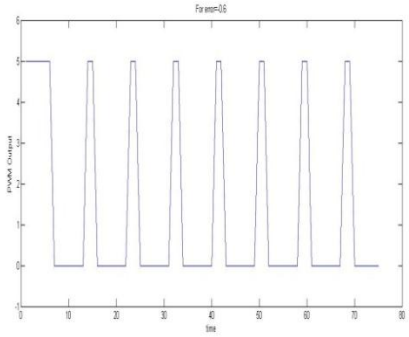
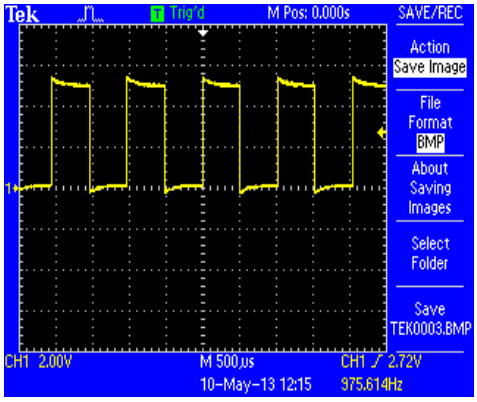
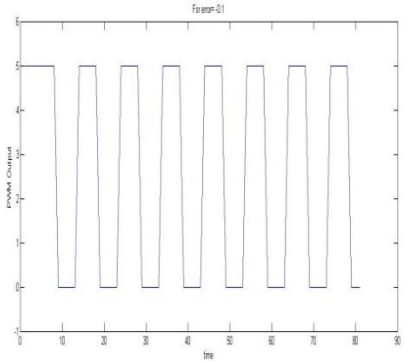
REFERENCE VALUE= 1.5V GIVEN VOLTAGE= 1.6V ERROR= -0.1 V		
REFERENCE VALUE= 1.5V GIVEN VOLTAGE= 2.1V ERROR= 0.6 V		
REFERENCE VALUE= 1.5V GIVEN VOLTAGE= 1.4V ERROR= -0.1 V		

TABLE 5.1- COMPARISON OF THE SRDUINO KIT AND SIMULINK OUTPUT FOR DIFFERENT ERROR SIGNALS

5.2 CONCLUSION:

From the comparison between the PWM outputs generated by the developed hardware (shown on the left) and the PWM outputs generated by the MATLAB-Simulink model, we verified the accuracy of our model. Besides, we observed that with the increase in error the pulse width increases implying the control output is given for more time.

CHAPTER 6

CONCLUSION AND FUTUTRE WORK

6.1 CONCLUSION

This work introduces a Simulink model for PID and FLC controllers and proves that FLC gives a better performance. By the application of FLC through the controller hardware designed via the ADC and the microcontroller, we tried to obtain a constant output voltage for a given input reference voltage corresponding to a set-point temperature. This result in an efficient controller for a system, and being microcontroller based, reduces the circuitry of the automated system to a great extent. Besides, the controller we designed is portable and cost-effective and can be used for a variety of plants with minor changes in the programming.

6.2 FUTURE WORK

The controller so designed can be applied to a specific plant and corresponding responses can be studied. Also the PWM output can be demodulated to give an analog control signal which could be used in case analog inputs are required for a plant.

REFERENCES

[1] C. J. Jiménez, S. Sánchez Solano, A. Barriga “Hardware Implementation of A General Purpose Fuzzy Controller”. Sixth International Fuzzy Systems Association World Congress (IFSA’95), Vol. 2, pp. 185-188, Sao Paulo - Brazil, July 21-28, 1995.

[2] Mudholkar R. R., Sawant S. R., 2002. “Fuzzy logic build estimation (FLTBE)”, Trans Indust Electron, IEEE, Vol.49, No.1,pp.264-267.

[3] Isizoh A. N., Okide S. O, Anazia A.E. OguC.D.,“Temperature Control System Using Fuzzy Logic Technique”, (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 1, No. 3, 2012. pp. 129-132.

[4] R.R. Mohler, "Nonlinear Systems: Volume I, Dynamics and Control," Prentice Hall, Englewood Cliffs, NJ, c1991.

[5] HU Yarryu, GUI Werhua,TANG Zhao-hui,TANG Ling, “Intelligenttemperature control system of quench furnace”,Trans.NonferrousMet.Soc.China,Vol 14,No 4. pp. 422-425.

[6] HE Jian-jun and YU Shou-yi, “Temperature Intelligent Control System of Large –Scale Standing Quench Furnace”. Journal of Electronic Science and Technology of China,Vol 3 ,No1. pp. 341-343.

[7] Kleanthis N., Costas N., and Christos S., 2006. “A comparison of classical, neural and fuzzy control for an underwater vehicle”,Proceedings of 7th WSEAS international conference on Neural Networks, pp.61-66.

[8] Lee S.H., Li Y.F. and Kapila V., 2004. “Development of a Matlab-Based Graphical User Interface for PIC Microcontroller Projects”, Proceedings of the American Society of Engineering Education Conference, Salt Lake City, UT, Session 2009.

[9] H. X. Li and H. B. Gatland, "A New Methodology forDesigning a Fuzzy Logic Controller", IEEE Transactions onSystems, Man, and Cybernetics, Vol. SMC-25, No. 3, pp.505-512, 1995.

[10] R. Isermann, "On Fuzzy Logic Applications for Automatic Control, Supervision, and Fault Diagnosis", IEEE Transactions on Systems, Man, and Cybernetics. Part A:Systems and Humans, Vol. SMC-28, No. 2, pp. 221-235,1998.

[11] "Using Atmel Studio for Arduino development". Megunolink.com. Retrieved 2013-01-18.

[12] Oxer, Jonathan; Blemings, Hugh (December 28, 2009). “Practical Arduino: Cool Projects for Open Source Hardware”. Edition 1. Apress. p. 450.

APPENDIX

PROGRAM CODE DEVELOPED IN C++

```
#include<iostream.h>
#include<conio.h>
int main()
{
inti,j;
floattemp,y;
floate,u;
cout<<"ENTER THE VALUES OF ERROR AND CHANGE IN ERROR: ";
cin>>e>>u;
float x0[3],x1[3],x2[3];
for(i=0;i<3;i++)
 {
x0[i]=0;
x1[i]=0;
x2[i]=0;
 }
if(e<-0.8)
x0[0]=-1.25*e-0.25;
if(e>-0.8&&e<-0.2)
 {
x0[0]=-1.25*e-0.25;
x0[1]=1.25*e+1;
 }
else if(e>-0.2&&e<0)
 {
x0[1]=1.25*e+1;
 }
else if(e>0&&e<0.2)
x0[1]=-1.25*e+1;
else if(e>0.2&&e<0.8)
 {
x0[1]=-1.25*e+1;
x0[2]=1.25*e-0.25;
 }
else if(e>0.8&&e<1)
x0[2]=1.25*e-0.25;
else
cout<<"INVALD!!!!";

if(u>-0.7 && u<-0.56)
x1[0]=-1.786*u-0.25;
if(u>-0.56 && u<-0.14)
{
x1[0]=-1.786*u-0.25;
x1[1]=1.786*u+1;
}
else if(u>-0.14 && u<0)
x1[1]=1.786*u+1;
else if(u>0 && u<0.14)
x1[1]=-1.786*u+1;
else if(u>0.14 && u<0.56)
{
x1[1]=-1.786*u+1;
x1[2]=1.786*u-0.25;
}
else if(u>0.56 && u<0.7)
x1[2]=1.786*u-0.25;
else
```

```
cout<<"INVALD!!!!";
cout<<"FUZZIFIED VALUES ARE: ";

for(i=0;i<3;i++)
cout<<"\t"<<x0[i]<<"\n";
for(i=0;i<3;i++)
cout<<"\t"<<x1[i]<<"\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
 {
if((x0[i]&&x1[j])!=0)
  {
if((i+j)<2)
   {
temp=x2[0];
x2[0]=x0[i]<x1[j]?x0[i]:x1[j];
x2[0]=temp>x2[0]?temp:x2[0];
   }
else if((i+j)==2)
   {
temp=x2[1];
x2[1]=x0[i]<x1[j]?x0[i]:x1[j];
x2[1]=temp>x2[1]?temp:x2[1];
   }
else if((i+j)>2)
   {
temp=x2[2];
x2[2]=x0[i]<x1[j]?x0[i]:x1[j];
x2[2]=temp>x2[2]?temp:x2[2];
   }
  }
 }
}

cout<<"OUTPUT FUZZY SET IS: \n";
for(i=0;i<3;i++)
cout<<x2[i]<<"\t";
 /*Defuzzification*/
 y=((-7*x2[0])+(0*x2[1])+(7*x2[2]))/(x2[0]+x2[1]+x2[2]);
cout<<"CRISP OUTPUT VALUE: "<<y;

getch();
return 0;
}
```

PROGRAM CODE DEVELOPED IN ARDUINO:

```
#define Rf 1.5
#define In  A3
#define Out 6
intRef,Val;  // Ref= Reference; Val= Actual sensor output
boolean i=false;
floater, cir;  // er=error; cir=change in error
floatdat[2]={0,0};  //storage variable for error values of t and t-1
float y;    //Crisp voltage Output
void setup()
{
Serial.begin(9600);
pinMode(Rf, INPUT);
pinMode(In, INPUT);
pinMode(Out,OUTPUT);
  Ref=Rf*1023/5;  //Read the reference value from A0 pin (analog input)
Serial.print("The reference Value = ");Serial.println(Ref);
}
void loop()
{
  ////analogWrite(Out,128);
  //while(1);
  Val=analogRead(In);
 //Read the sensor value from A0 pin (analog input)
Serial.print("The actual Value = ");
Serial.println((float)(Val*5)/1023);
er=(float)(Ref-Val);er=(er*5)/1023;
 //Compute the error by comparing both values
Serial.print("The Error Value is = ");
Serial.println(er);
dat[i]=er;
//Store the error value
cir=dat[i]-dat[!i];
//Compute the change in error, i.e, present-past
  i=!i;
Serial.print("The Change inError Value is = ");
Serial.println(cir);
  y=fuzzify(er,cir);
//Obtain the crisp value from fuzzy logic controller
  y=(y+7)*255/14;
analogWrite(Out,(int)y);
//Send the PWM output
Serial.print("The Crisp Value is = ");
Serial.println((int)y);
delay(500);
  //while(1);
}
floatfuzzify(float e,float u)
{
float x0[3],x1[3],x2[3],y,temp;
for(int i=0;i<3;i++)
    {
x0[i]=0;
x1[i]=0;
x2[i]=0;
    }
if(e<-0.8)
    {
x0[0]=-1.25*e-0.25;}
```

```
if((e>-0.8) && (e<-0.2))
    {
x0[0]=-1.25*e-0.25;
x0[1]=1.25*e+1;
    }
else if((e>-0.2) && (e<0))
    {
x0[1]=1.25*e+1;
    }
else if((e>0)&&(e<0.2))
    {
x0[1]=-1.25*e+1;
    }
else if((e>0.2) && (e<0.8))
    {
x0[1]=-1.25*e+1;
x0[2]=1.25*e-0.25;
    }
else if(e>0.8) //&& (e<1))
    {
x0[2]=1.25*e-0.25;
    }
    //else
   //cout<<"INVALD!!!!";

if((u>-0.7) && (u<-0.56))
    {
x1[0]=-1.786*u-0.25;
    }
if((u>-0.56) && (u<-0.14))
    {
x1[0]=-1.786*u-0.25;
x1[1]=1.786*u+1;
    }
else if((u>-0.14) && (u<=0))
    {
x1[1]=1.786*u+1;
    }
else if((u>0) && (u<0.14))
    {
x1[1]=-1.786*u+1;
    }
else if((u>0.14) && (u<0.56))
    {
x1[1]=-1.786*u+1;
x1[2]=1.786*u-0.25;
    }
else if((u>0.56) && (u<0.7))
    {
x1[2]=1.786*u-0.25;
    }
    //else
    //cout<<"INVALD!!!!";
    //cout<<"FUZZIFIED VALUES ARE: ";
    //for(i=0;i<3;i++)
    //cout<<"\t"<<x0[i]<<"\n";
    //for(i=0;i<3;i++)
    //cout<<"\t"<<x1[i]<<"\n";
for(int i=0;i<3;i++)
    {
for(int j=0;j<3;j++)
```

```
       {
if((x0[i]!=0 )&& (x1[j]!=0))
          {
if((i+j)<2)
              {
temp=x2[0];
x2[0]=x0[i]<x1[j]?x0[i]:x1[j];
x2[0]=temp>x2[0]?temp:x2[0];
              }
else if((i+j)==2)
              {
temp=x2[1];
x2[1]=x0[i]<x1[j]?x0[i]:x1[j];
x2[1]=temp>x2[1]?temp:x2[1];
              }
else if((i+j)>2)
              {
temp=x2[2];
x2[2]=x0[i]<x1[j]?x0[i]:x1[j];
x2[2]=temp>x2[2]?temp:x2[2];
              }
          }
       }
   }
   //cout<<"OUTPUT FUZZY SET IS: \n";
   //for(i=0;i<3;i++)
   //cout<<x2[i]<<"\t";
  /*Defuzzification*/
   y=((-7*x2[0])+(0*x2[1])+(7*x2[2]))/(x2[0]+x2[1]+x2[2]);
   //cout<<"CRISP OUTPUT VALUE: "<<y;
return y;
}
```